

Stream processing with R in AWS

AWR, AWR.KMS, AWR.Kinesis (R packages) used in ECS



CARD.COM

Gergely Daroczi

@daroczig

March 14, 2017







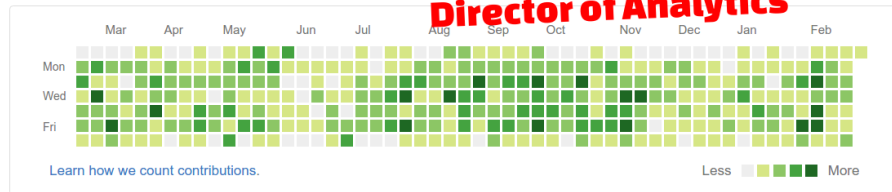
3,633 contributions in the year before last

Lead R Developer



3,470 contributions in the last year

Director of Analytics





Gergely Daróczy @daroczig · Apr 11

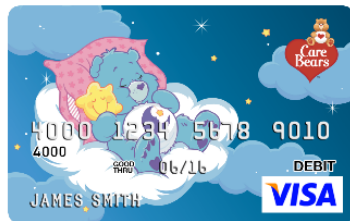
Just received my "I ❤️ R" prepaid debit card from @CARD. Will be fun to use this #rstats designed card at #user2015 :)

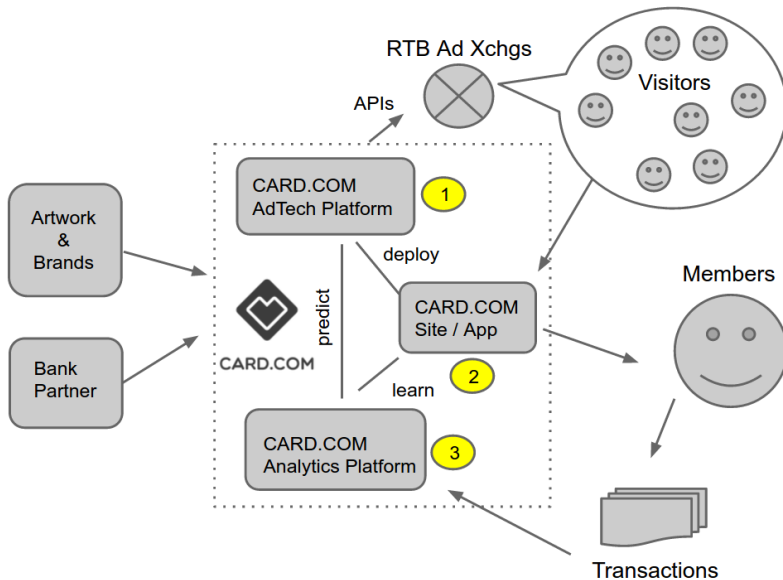


RETWEETS
10

FAVORITES
16







- card transaction processors
- card manufacturers
- CIP/KYC service providers
- online ad platforms
- remarketing networks
- licensing partners
- communication engines
- others





Classes and code fragments:

- Classes
- Code Snippets
- Input fields
 - Getting fields...please wait
- Info fields
 - Getting fields...please wait
- Output fields
 - Getting fields...please wait

Step name

Shorten Date

Class code

```

@ Processor
import java.text.SimpleDateFormat;
import java.util.Date;
import java.text.ParseException;
import java.util.TimeZone;

private SimpleDateFormat df1 = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss.SSS");
private SimpleDateFormat df2 = new SimpleDateFormat("yyyy-MM-dd HH");

public boolean processRow(StepMetaInterface smi, StepDataInterface sdi) throws KettleException, ParseException
{
    Object[] r = getRow();
    if (r == null) {
        setOutputDone();
        return false;
    }

    if (first)
    {
        first = false;
    }

    // It is always safest to call createOutputRow() to ensure that your output row's Object[] is large
    // enough to handle any new fields you are creating in this step.
    r = createOutputRow(r, data.outputRowMeta.size());

    df2.setTimeZone(TimeZone.getTimeZone("America/Los_Angeles"));

```

Line #: 0

Fields Parameters Info steps Target steps

Fields

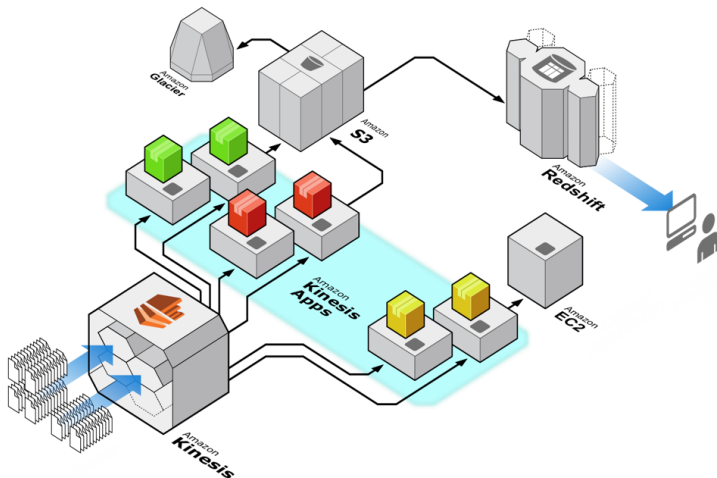
Clear the result fields?

#	Fieldname	Type	Length	Precision
1	RPT_DATE_SHORT	String		

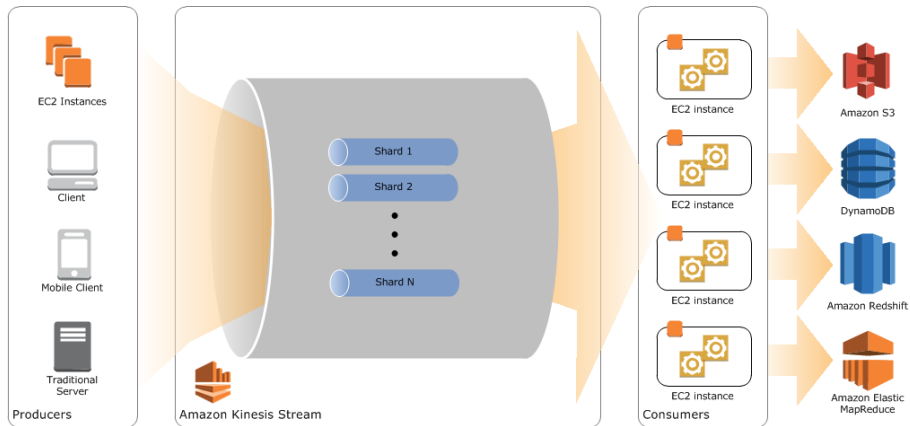
Help

OK Cancel Test class

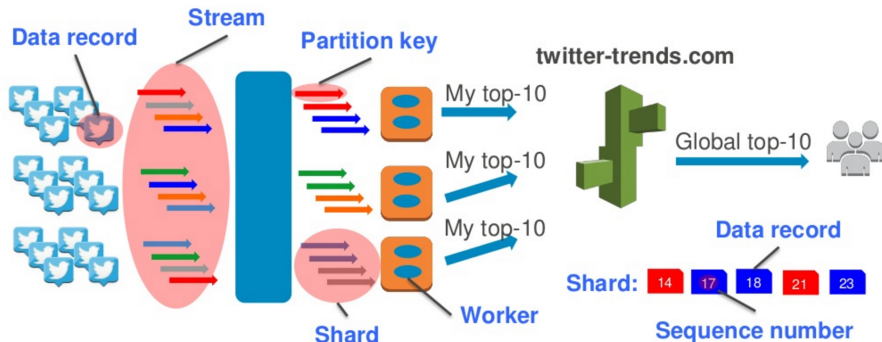
Why Amazon Kinesis?



Source: [Kinesis Product Details](#)

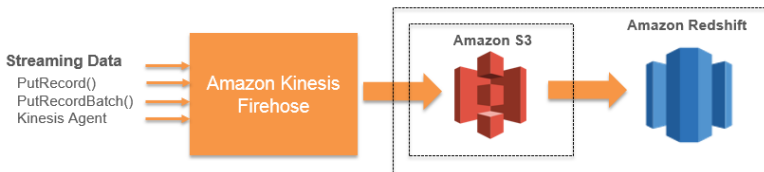


Source: [Kinesis Developer Guide](#)



Source: AWS re:Invent 2013

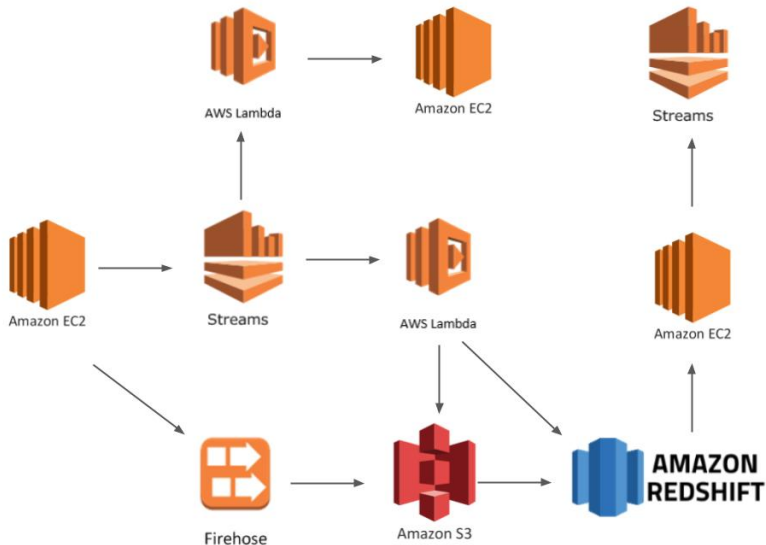




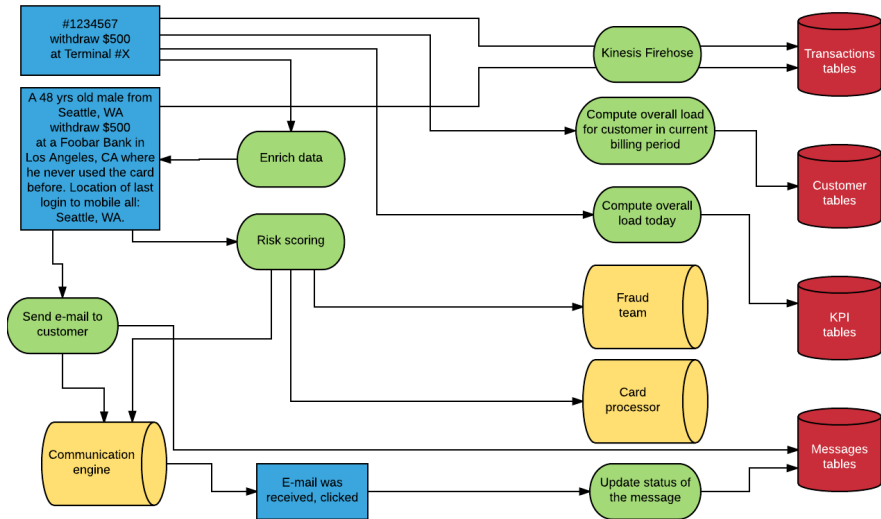
```
> x <- 3.14
> attr(x, 'class') <- 'standard'

> print.standard <- function(x, ...) {
+   ## SLA
+   if (runif(1) * 100 > 99.9) {
+     Sys.sleep(20)
+   }
+   futile.logger::flog.info(x)
+ }

> while (TRUE) print(x)
INFO [2017-03-03 22:27:57] 3.14
INFO [2017-03-03 22:27:57] 3.14
INFO [2017-03-03 22:27:57] 3.14
INFO [2017-03-03 22:28:17] 3.14
INFO [2017-03-03 22:28:17] 3.14
```



Example use-case



Writing data to the stream:

- Amazon Kinesis Streams API, SDK
- Amazon Kinesis Producer Library (KPL) from Java
- flume-kinesis
- Amazon Kinesis Agent

Reading data from the stream:

- Amazon Kinesis Streams API, SDK
- Amazon Kinesis Client Library (KCL) from Java, Node.js, .NET, Python, Ruby

Managing streams:

- Amazon Kinesis Streams API (!)

```
> library(rJava)
> .jinit(classpath = list.files('~/.Projects/AWR/inst/java/', full.names = TRUE))

> kc <- .jnew('com.amazonaws.services.kinesis.AmazonKinesisClient')
> kc$setEndpoint('kinesis.us-west-2.amazonaws.com', 'kinesis', 'us-west-2')

> sir <- .jnew('com.amazonaws.services.kinesis.model.GetShardIteratorRequest')
> sir$setStreamName('test_kinesis')
> sir$setShardId(.jnew('java/lang/String', '0'))
> sir$setShardIteratorType('TRIM_HORIZON')
> iterator <- kc$getShardIterator(sir)$getShardIterator()

> grr <- .jnew('com.amazonaws.services.kinesis.model.GetRecordsRequest')
> grr$setShardIterator(iterator)
> kc$getRecords(grr)$getRecords()
[1] "Java-Object{[{SequenceNumber: 495628941604494443321533463710843135723243616650,
ApproximateArrivalTimestamp: Tue Jun 14 09:40:19 CEST 2016,
Data: java.nio.HeapByteBuffer[pos=0 lim=6 cap=6],PartitionKey: 42}]}"

> sapply(kc$getRecords(grr)$getRecords(),
+        function(x)
+        rawToChar(x$getData()$array()))
[1] "foobar"
```

```
> ms <- .jnew('com.amazonaws.services.kinesis.model.MergeShardsRequest')
> ms$setShardToMerge('shardId-000000000000')
> ms$setAdjacentShardToMerge('shardId-000000000001')
> ms$setStreamName('test_kinesis')
> kc$mergeShards(ms)
```

```

> kc$describeStream(StreamName = 'test_kinesis')$getStreamDescription()$getShards()
[1] "Java-Object{[
{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey: 170
SequenceNumberRange: {
StartingSequenceNumber: 49562894160427143586954815717376297430913467927668719618,
EndingSequenceNumber: 49562894160438293959554081028945856364232263390243848194}},
{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey: 17014118346046923173
SequenceNumberRange: {
StartingSequenceNumber: 49562894160449444332153346340517833149186116289174700050,
EndingSequenceNumber: 49562894160460594704752611652087392082504911751749828626}},
{ShardId: shardId-000000000002,
ParentShardId: shardId-000000000000,
AdjacentParentShardId: shardId-000000000001,
HashKeyRange: {StartingHashKey: 0,EndingHashKey: 3402823669209384634633746074317682
SequenceNumberRange: {StartingSequenceNumber: 4956290499149767309970492434472701952

```

- An *easy-to-use* programming model for processing data

```
java -cp amazon-kinesis-client-1.7.3.jar \  
com.amazonaws.services.kinesis.multilang.MultiLangDaemon \  
app.properties
```

- *Scalable* and *fault-tolerant* processing (checkpointing via DynamoDB)
- Logging and metrics in CloudWatch
- The **MultiLangDaemon** spawns processes written in any language, communication happens via JSON messages sent over stdin/stdout
- Only a few events/methods to care about in the consumer application:
 - 1 initialize
 - 2 processRecords
 - 3 checkpoint
 - 4 shutdown

① **initialize:**

- Perform initialization steps
- Write “status” message to indicate you are done
- Begin reading line from STDIN to receive next action

② **processRecords:**

- Perform processing tasks (you may write a checkpoint message at any time)
- Write “status” message to STDOUT to indicate you are done.
- Begin reading line from STDIN to receive next action

③ **shutdown:**

- Perform shutdown tasks (you may write a checkpoint message at any time)
- Write “status” message to STDOUT to indicate you are done.
- Begin reading line from STDIN to receive next action

④ **checkpoint:**

- Decide whether to checkpoint again based on whether there is an error or not.

Classes and code fragments:

- Classes
- Code Snippets
- Input fields
 - Getting fields...please wait
- Info fields
 - Getting fields...please wait
- Output fields
 - Getting fields...please wait

Step name

Shorten Date

Class code

```

@ Processor
import java.text.SimpleDateFormat;
import java.util.Date;
import java.text.ParseException;
import java.util.TimeZone;

private SimpleDateFormat df1 = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss.SSS");
private SimpleDateFormat df2 = new SimpleDateFormat("yyyy-MM-dd HH");

public boolean processRow(StepMetaInterface smi, StepDataInterface sdi) throws KettleException, ParseException
{
    Object[] r = getRow();
    if (r == null) {
        setOutputDone();
        return false;
    }

    if (first)
    {
        first = false;
    }

    // It is always safest to call createOutputRow() to ensure that your output row's Object[] is large
    // enough to handle any new fields you are creating in this step.
    r = createOutputRow(r, data.outputRowMeta.size());

    df2.setTimeZone(TimeZone.getTimeZone("America/Los_Angeles"));

```

Line #: 0

Fields Parameters Info steps Target steps

Fields

Clear the result fields?

#	Fieldname	Type	Length	Precision
1	RPT_DATE_SHORT	String		

Help

OK Cancel Test class

```
#!/usr/bin/r -i

while (TRUE) {

  ## read and parse JSON messages
  line <- fromJSON(readLines(n = 1))

  ## nothing to do unless we receive records to process
  if (line$action == 'processRecords') {

    ## process each record
    lapply(line$records, function(r) {

      business_logic(fromJSON(rawToChar(base64_dec(r$data))))
      cat(toJSON(list(action = 'checkpoint', checkpoint = r$sequenceNumber)))

    })
  }

  ## return response in JSON
  cat(toJSON(list(action = 'status', responseFor = line$action)))
}
```

```
#!/usr/bin/r -i
```

```
while (TRUE) {
```

```
## read and parse JSON messages
```

```
line <- fromJSON(readLines(n = 1))
```

```
## nothing to do unless we receive records to process
```

```
if (line$action == 'processRecords') {
```

```
## process each record
```

```
lapply(line$records, function(r) {
```

```
    business_logic(fromJSON(rawToChar(base64_dec(r$data))))
```

```
    cat(toJSON(list(action = 'checkpoint', checkpoint = r$sequenceNumber)))
```

```
  })
```

```
}
```

```
## return response in JSON
```

```
cat(toJSON(list(action = 'status', responseFor = line$action)))
```

```
}
```

```
> install.packages('AWR.Kinesis')
also installing the dependency 'AWR'

trying URL 'https://cloud.r-project.org/src/contrib/AWR_1.11.89.tar.gz'
Content type 'application/x-gzip' length 3125 bytes

trying URL 'https://cloud.r-project.org/src/contrib/AWR.Kinesis_1.7.3.tar.gz'
Content type 'application/x-gzip' length 3091459 bytes (2.9 MB)

* installing *source* package 'AWR' ...
** testing if installed package can be loaded
trying URL 'https://gitlab.com/cardcorp/AWR/repository/archive.zip?ref=1.11.89'
downloaded 58.9 MB
* DONE (AWR)

* installing *source* package 'AWR.Kinesis' ...
* DONE (AWR.Kinesis)
```

Business logic coded in R (demo_app.R):

```
library(AWR.Kinesis)
kinesis_consumer(processRecords = function(records) {
  flog.info(jsonlite::toJSON(records))
})
```

Business logic coded in R (demo_app.R):

```
library(AWR.Kinesis)
kinesis_consumer(processRecords = function(records) {
  flog.info(jsonlite::toJSON(records))
})
```

Note

This is not something you should run in RStudio.

Business logic coded in R (demo_app.R):

```
library(AWR.Kinesis)
kinesis_consumer(processRecords = function(records) {
  flog.info(jsonlite::toJSON(records))
})
```

Config file for the MultiLangDaemon (demo_app.properties):

```
executableName = ./demo_app.R
streamName = demo_stream
applicationName = demo_app
```

Start the MultiLangDaemon:

```
/usr/bin/java -cp AWR/java/*:AWR.Kinesis/java/*:/ \
  com.amazonaws.services.kinesis.multilang.MultiLangDaemon \
  ./demo_app.properties
```

```
library(futile.logger)
library(AWR.Kinesis)

kinesis_consumer(

  initialize      = function()
    flog.info('Hello'),

  processRecords = function(records)
    flog.info(paste('Received', nrow(records), 'records from Kinesis')),

  shutdown       = function()
    flog.info('Bye'),

  updater        = list(
    list(1, function()
      flog.info('Updating some data every minute')),
    list(1/60*10, function()
      flog.info(paste(
        'This is a high frequency updater call',
        'running every 10 seconds')))),

  checkpointing = 1,
  logfile       = '/logs/logger.log')
```

Note

In theory you could, but this is not something you should run in RStudio.

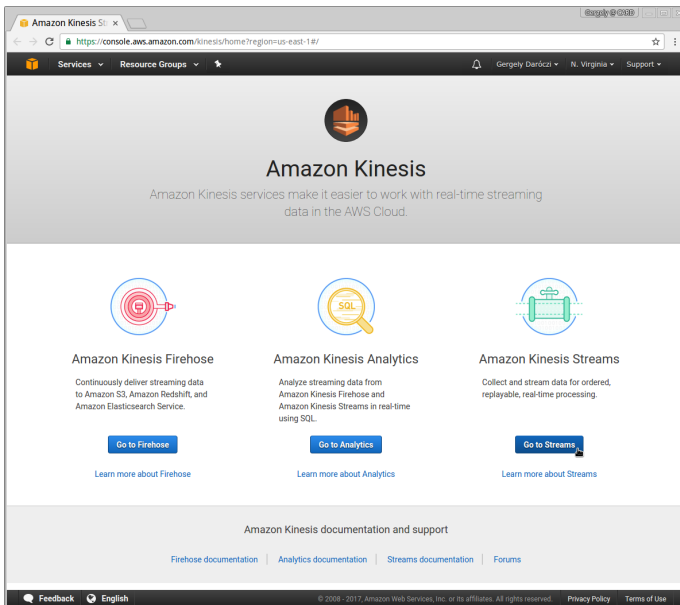
- 1 Create a Kinesis Stream
- 2 Create an IAM user with DynamoDB and Kinesis permissions
- 3 Write data to the Stream
- 4 Run the MultiLangDaemon referencing the properties file

Note

In theory you could, but this is not something you should run in RStudio.

- 1 Create a Kinesis Stream
- 2 Create an IAM user with DynamoDB and Kinesis permissions
- 3 Write data to the Stream
- 4 Run the MultiLangDaemon referencing the properties file





The screenshot shows the Amazon Kinesis console interface. At the top, there's a navigation bar with 'Services', 'Resource Groups', and user information. The main header features the Amazon Kinesis logo and a brief description: 'Amazon Kinesis services make it easier to work with real-time streaming data in the AWS Cloud.' Below this, three service cards are displayed: 'Amazon Kinesis Firehose' (with a red icon), 'Amazon Kinesis Analytics' (with a yellow icon), and 'Amazon Kinesis Streams' (with a green icon). Each card includes a short description, a 'Go to [Service]' button, and a link to 'Learn more about [Service]'. The bottom section provides links to 'Amazon Kinesis documentation and support', including 'Firehose documentation', 'Analytics documentation', 'Streams documentation', and 'Forums'. The footer contains a feedback link, language selector, and copyright information.

Amazon Kinesis

Amazon Kinesis services make it easier to work with real-time streaming data in the AWS Cloud.

Amazon Kinesis Firehose

Continuously deliver streaming data to Amazon S3, Amazon Redshift, and Amazon Elasticsearch Service.

[Go to Firehose](#)

[Learn more about Firehose](#)

Amazon Kinesis Analytics

Analyze streaming data from Amazon Kinesis Firehose and Amazon Kinesis Streams in real-time using SQL.

[Go to Analytics](#)

[Learn more about Analytics](#)

Amazon Kinesis Streams

Collect and stream data for ordered, replayable, real-time processing.

[Go to Streams](#)

[Learn more about Streams](#)

Amazon Kinesis documentation and support

[Firehose documentation](#) | [Analytics documentation](#) | [Streams documentation](#) | [Forums](#)

[Feedback](#) [English](#)

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

Amazon Kinesis Streams

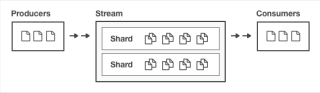
Streams
Firehose
Analytics

Create stream

Stream name* test-AWR

Shards

A shard is a unit of throughput capacity. Each shard ingests up to 1MB/sec and 1000 records/sec, and emits up to 2MB/sec. To accommodate for higher or lower throughput, the number of shards can be modified after the stream is created using the API. [Learn more](#)



```
graph LR; Producers[Producers] --> Stream[Stream]; subgraph Stream; direction TB; S1[Shard]; S2[Shard]; end; Stream --> Consumers[Consumers];
```

Estimate the number of shards you'll need

Number of shards* 1

You can provision up to 48 more shards before hitting your account limit of 50. [Learn more](#) or [request a shard limit increase for this account](#)

Total stream capacity Values are calculated based on the number of shards entered above.

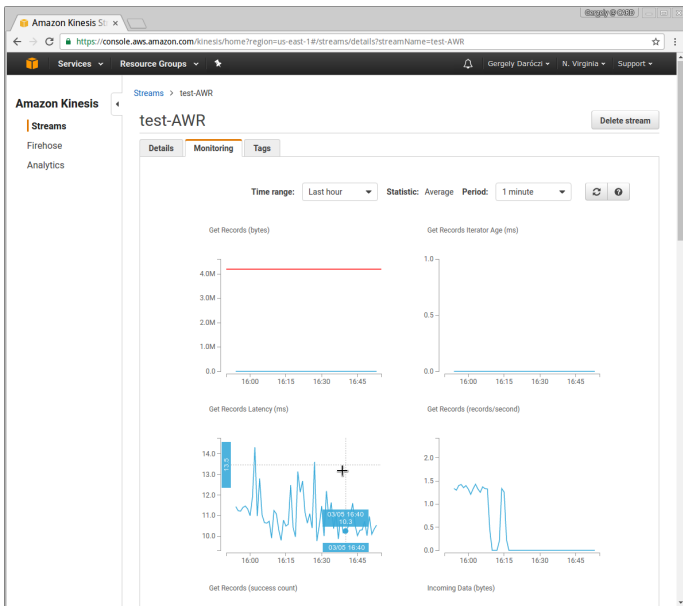
Write 1 MB per second

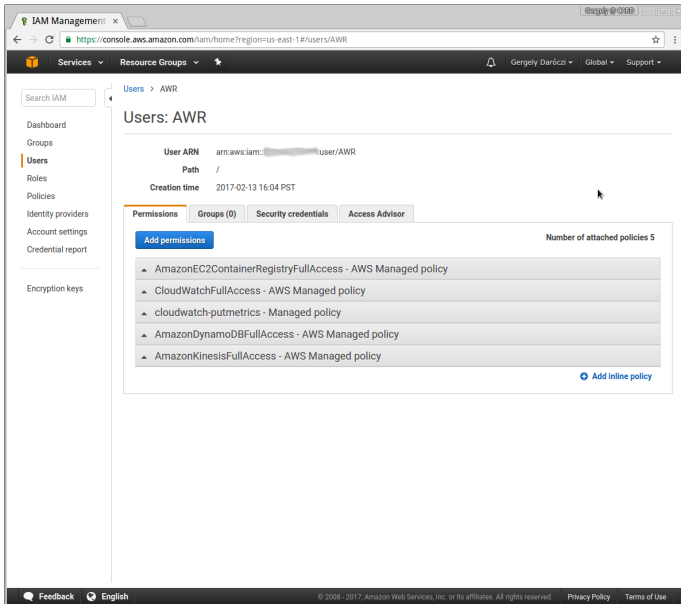
1000 Records per second

Read 2 MB per second

* Required

Cancel Create stream





The screenshot shows the AWS IAM console interface. The browser address bar displays `https://console.aws.amazon.com/iam/home?region=us-east-1#/users/AWR`. The top navigation bar includes 'Services', 'Resource Groups', and user information 'Gergely Daróczi'. A left-hand sidebar contains navigation links: 'Dashboard', 'Groups', 'Users' (highlighted), 'Roles', 'Policies', 'Identity providers', 'Account settings', 'Credential report', and 'Encryption keys'. The main content area is titled 'Users: AWR' and displays the following details:

- User ARN:** `arn:aws:iam::[redacted]:user/AWR`
- Path:** `/`
- Creation time:** 2017-02-13 16:04 PST

Below these details are four tabs: 'Permissions' (selected), 'Groups (0)', 'Security credentials', and 'Access Advisor'. Under the 'Permissions' tab, there is a blue 'Add permissions' button and a text label 'Number of attached policies 5'. A list of five attached policies is shown, each with a triangle icon:

- AmazonEC2ContainerRegistryFullAccess - AWS Managed policy
- CloudWatchFullAccess - AWS Managed policy
- cloudwatch-putmetrics - Managed policy
- AmazonDynamoDBFullAccess - AWS Managed policy
- AmazonKinesisFullAccess - AWS Managed policy

A blue link 'Add inline policy' is located at the bottom right of the policy list. The footer of the console includes 'Feedback', 'English', copyright information '© 2008 - 2017 Amazon Web Services, Inc. or its affiliates. All rights reserved.', and links to 'Privacy Policy' and 'Terms of Use'.

```
library(rJava)
.jcall("java/lang/System", "S", "setProperty", "aws.profile", "personal")

library(AWR.Kinesis)
library(jsonlite)
library(futile.logger)
library(nycflights13)
while (TRUE) {

  ## pick a ~car~flight
  flight <- flights[sample(1:nrow(flights), 1), ]

  ## prr <- .jnew('com.amazonaws.services.kinesis.model.PutRecordRequest')
  ## prr$setStreamName('test1')
  ## prr$setData(J('java.nio.ByteBuffer')$wrap(.jbyte(charToRaw(toJSON(car)))))
  ## prr$setPartitionKey(rownames(car))
  ## kc$putRecord(prr)

  res <- kinesis_put_record(stream = 'test-AWR', region = 'us-east-1',
                           data = toJSON(flight), partitionKey = flight$dest)
  flog.info(paste('Pushed a new flight to Kinesis:', res$sequenceNumber))

}
```

Write Data to the Stream from R

```
File Edit Options Buffers Tools Minibuf Help
*Minibuf-1*

library(RJava)
.jcall("java.lang.System", "S", "setProperty", "aws_profile", "personal")
library(AWR.Kinesis): library(Jsonlite); library(futile.logger): library(nycflights13)
while (TRUE) {
  flight <- flights[sample(nrow(flights), 1), ]
  res <- kinesis_put_record(test_AWR, region = "us-east-1", data = toJSON(flight),
                           partitionKey = flight$dest)
  flog.info(paste("Pushed a new flight to Kinesis:", res$sequenceNumber))
}

INFO [2017-03-06 21:47:16] Pushed a new flight to Kinesis: 49571082550613725758991014186133813750336290428328869938
INFO [2017-03-06 21:47:17] Pushed a new flight to Kinesis: 49571082550613725758991014186380434617537674848688406578
INFO [2017-03-06 21:47:17] Pushed a new flight to Kinesis: 495710825506137257589910141866103052326454391882580018
INFO [2017-03-06 21:47:17] Pushed a new flight to Kinesis: 4957108255061372575899101418687004957448159964444407858
INFO [2017-03-06 21:47:20] Pushed a new flight to Kinesis: 495710825506137257589910141807534061663730515784895538
INFO [2017-03-06 21:47:18] Pushed a new flight to Kinesis: 495710825506137257589910141873947238019434879486364978
INFO [2017-03-06 21:47:19] Pushed a new flight to Kinesis: 49571082550613725758991014187636508544117274698647076914
INFO [2017-03-06 21:47:19] Pushed a new flight to Kinesis: 49571082550613725758991014187857741969106751837618307122
INFO [2017-03-06 21:47:20] Pushed a new flight to Kinesis: 495710825506137257589910141800156920428756672442577202
INFO [2017-03-06 21:47:20] Pushed a new flight to Kinesis: 495710825506137257589910141882550642860481506388117291058
INFO [2017-03-06 21:47:21] Pushed a new flight to Kinesis: 49571082550613725758991014188505726208420193143980294194
INFO [2017-03-06 21:47:21] Pushed a new flight to Kinesis: 49571082550613725758991014188814002292421923652249845810
INFO [2017-03-06 21:47:21] Pushed a new flight to Kinesis: 495710825506137257589910141890156920428756672442577202
INFO [2017-03-06 21:47:22] Pushed a new flight to Kinesis: 49571082550613725758991014189266140548957795032309432370
INFO [2017-03-06 21:47:22] Pushed a new flight to Kinesis: 49571082550613725758991014189516388193618023271473610802
INFO [2017-03-06 21:47:23] Pushed a new flight to Kinesis: 49571082550613725758991014189794441132129388119094984754
INFO [2017-03-06 21:47:23] Pushed a new flight to Kinesis: 4957108255061372575899101419001567455711868258066214062
INFO [2017-03-06 21:47:24] Pushed a new flight to Kinesis: 49571082550613725758991014190342084528414815135236882482
INFO [2017-03-06 21:47:24] Pushed a new flight to Kinesis: 49571082550613725758991014190545184066110072905306996786
INFO [2017-03-06 21:47:25] Pushed a new flight to Kinesis: 49571082550613725758991014190772462120197623190151757874
INFO [2017-03-06 21:47:25] Pushed a new flight to Kinesis: 49571082550613725758991014190696902513353638810746034
INFO [2017-03-06 21:47:25] Pushed a new flight to Kinesis: 49571082550613725758991014191170198714850636257249566514
INFO [2017-03-06 21:47:26] Pushed a new flight to Kinesis: 4957108255061372575899101419139747676893836542194327602
INFO [2017-03-06 21:47:26] Pushed a new flight to Kinesis: 49571082550613725758991014191629590526304395412457390130
INFO [2017-03-06 21:47:27] Pushed a new flight to Kinesis: 4957108255061372575899101419191006131645489388092822962
INFO [2017-03-06 21:47:27] Pushed a new flight to Kinesis: 495710825506137257589910141921739571742250168948700210
INFO [2017-03-06 21:47:28] Pushed a new flight to Kinesis: 49571082550613725758991014192373079905367392423621165106
INFO [2017-03-06 21:47:28] Pushed a new flight to Kinesis: 49571082550613725758991014192595522256176484191767101490
INFO [2017-03-06 21:47:28] Pushed a new flight to Kinesis: 4957108255061372575899101419277202542584020119993679922
INFO [2017-03-06 21:47:29] Pushed a new flight to Kinesis: 4957108255061372575899101419298116092643355067217648370
INFO [2017-03-06 21:47:29] Pushed a new flight to Kinesis: 49571082550613725758991014193206029795081871993713197106
INFO [2017-03-06 21:47:30] Pushed a new flight to Kinesis: 4957108255061372575899101419341396503605558280482136114
INFO [2017-03-06 21:47:30] Pushed a new flight to Kinesis: 4957108255061372575899101419360981101883315206704536626
INFO [2017-03-06 21:47:30] Pushed a new flight to Kinesis: 495710825506137257589910141938189551852648912278181810
INFO [2017-03-06 21:47:31] Pushed a new flight to Kinesis: 49571082550613725758991014194043815388074810149223530546
INFO [2017-03-06 21:47:31] Pushed a new flight to Kinesis: 49571082550613725758991014194267466664703516546544173106
INFO [2017-03-06 21:47:32] Pushed a new flight to Kinesis: 49571082550613725758991014194544310677395266960271364146
INFO [2017-03-06 21:47:32] Pushed a new flight to Kinesis: 49571082550613725758991014194752245918368982914320826418
INFO [2017-03-06 21:47:33] Pushed a new flight to Kinesis: 49571082550613725758991014195169325326136030048313938374
INFO [2017-03-06 21:47:34] Pushed a new flight to Kinesis: 49571082550613725758991014195478810335957375117038714930
INFO [2017-03-06 21:47:34] Pushed a new flight to Kinesis: 49571082550613725758991014195897098669444036880206528562
INFO [2017-03-06 21:47:35] Pushed a new flight to Kinesis: 495710825506137257589910141962162508522229892238859926
INFO [2017-03-06 21:47:35] Pushed a new flight to Kinesis: 49571082550613725758991014196406056439601795831747305394
INFO [2017-03-06 21:47:36] Pushed a new flight to Kinesis: 49571082550613725758991014196589813164183219466032644146
INFO [2017-03-06 21:47:36] Pushed a new flight to Kinesis: 4957108255061372575899101419681104658917269667372351090
INFO [2017-03-06 21:47:36] Pushed a new flight to Kinesis: 495710825506137257589910141910068925719592666002515602
INFO [2017-03-06 21:47:37] Pushed a new flight to Kinesis: 49571082550613725758991014197254722364971265580840517682
INFO [2017-03-06 21:47:37] Pushed a new flight to Kinesis: 4957108255061372575899101419747595789960742857250701362
INFO [2017-03-06 21:47:38] Pushed a new flight to Kinesis: 4957108255061372575899101419724994508801356467240173618
INFO [2017-03-06 21:47:38] Pushed a new flight to Kinesis: 49571082550613725758991014191969701460046747472619278
INFO [2017-03-06 21:47:39] Pushed a new flight to Kinesis: 49571082550613725758991014196145700694027247488757399602
INFO [2017-03-06 21:47:39] Pushed a new flight to Kinesis: 495710825506137257589910141984128733001620805363674644498
INFO [2017-03-06 21:47:40] Pushed a new flight to Kinesis: 495710825506137257589910141964051135424930809931702322
INFO [2017-03-06 21:47:40] Pushed a new flight to Kinesis: 4957108255061372575899101419925417587222621172722389
INFO [2017-03-06 21:47:41] Pushed a new flight to Kinesis: 49571082550613725758991014199624216971415939106862006322
* 12M U: R-kinesis-putrecord LESS [R]: run
x butterfly
```

```
## get an iterator
sir <- .jnew('com.amazonaws.services.kinesis.model.GetShardIteratorRequest')
sir$setStreamName('test-AWR')
sir$setShardId(.jnew('java/lang/String', '0'))
sir$setShardIteratorType('TRIM_HORIZON')
kc <- .jnew('com.amazonaws.services.kinesis.AmazonKinesisClient')
kc$setEndpoint('kinesis.us-east-1.amazonaws.com')
iterator <- kc$getShardIterator(sir)$getShardIterator()

## get records
grr <- .jnew('com.amazonaws.services.kinesis.model.GetRecordsRequest')
grr$setShardIterator(iterator)
records <- kc$getRecords(grr)$getRecords()

## transform to string
json <- sapply(records, function(x)
  rawToChar(x$getData()$array()))

## decode JSON
json[1]
fromJSON(json[1])
rbindlist(lapply(json, fromJSON))
```

Running the MultiLangDaemon locally

```
Terminix: Default
1: daroczi@gergely-CARD: ~/Projects/card-rocker/r-kinesis-example/files
~/Projects/card-rocker/r-kinesis-example/files master ? export AWS_PROFILE=personal
~/Projects/card-rocker/r-kinesis-example/files master ? /usr/bin/java -cp \
"/usr/local/lib/R/site-library/AWR/java/*:/usr/local/lib/R/site-library/AWR.Kinesis/java/*:/usr/local/lib/R/site-library/AWR.KinesisClientLibConfigurator.jar" \
com.amazonaws.services.kinesis.multipLang.MultiLangDaemon ./app.properties
Mar 05, 2017 5:32:33 PM com.amazonaws.services.kinesis.clientlibrary.config.KinesisClientLibConfigurator getConfiguration
INFO: Value of workerId is not provided in the properties. WorkerId is automatically assigned as:
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.clientlibrary.config.KinesisClientLibConfigurator withProperty
INFO: Successfully set property regionName with value us-east-1
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.multipLang.MultiLangDaemonConfig buildExecutorService
INFO: Using a cached thread pool.
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.multipLang.MultiLangDaemonConfig <init>
INFO: Running AWR-demo-app to process stream test-AWR with executable /app/app.R
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.multipLang.MultiLangDaemonConfig prepare
INFO: Using workerId:
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.multipLang.MultiLangDaemonConfig prepare
INFO: Using credentials with access key id:
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.multipLang.MultiLangDaemonConfig prepare
INFO: MultiLangDaemon is adding the following fields to the User Agent: amazon-kinesis-client-library-java-1.7.3 amazon-kinesis-multi-lang-daemon/1.0.1 R /app/app.R
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.leases.impl.LeaseCoordinator <init>
INFO: With fallover time 10000 ms and epsilon 25 ms, LeaseCoordinator will renew leases every 3308 ms, takeleases every 20050 ms, process maximum of 2147483647 leases and steal 1 lease(s) at a time.
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker initialize
INFO: Initialization attempt 1
Mar 05, 2017 5:32:34 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker initialize
INFO: Initializing LeaseCoordinator
Mar 05, 2017 5:32:35 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker initialize
INFO: Syncing Kinesis shard info
Mar 05, 2017 5:32:36 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker initialize
INFO: Starting LeaseCoordinator
Mar 05, 2017 5:32:36 PM com.amazonaws.services.kinesis.leases.impl.LeaseTaker computeLeasesToTake
INFO: Worker needed 2 leases but none were expired, so it will steal lease shardId-0000000000002 from bef54447-3adb-444f-8d0c-67504e5c86ef
Mar 05, 2017 5:32:36 PM com.amazonaws.services.kinesis.leases.impl.LeaseTaker computeLeasesToTake
INFO: Worker saw 3 total leases, 0 available leases, 2 workers. Target is 2 leases, I have 0 leases, I will take 1 leases
Mar 05, 2017 5:32:36 PM com.amazonaws.services.kinesis.leases.impl.LeaseTaker takeLeases
INFO: Worker successfully took 1 leases: shardId-0000000000002
Mar 05, 2017 5:32:46 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker run
INFO: Initialization complete. Starting worker loop.
Mar 05, 2017 5:32:46 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker infoForce
INFO: Created new shardConsumer for : ShardInfo [shardId=shardId-0000000000002, concurrencyToken=, parentShardId=shardId-0000000000000], checkpoint={SequenceNumber: TRIM_HORIZON, SubsequenceNumber: 0}]
Mar 05, 2017 5:32:46 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.BlockOnParentShardTask call
INFO: No need to block on parents [shardId-0000000000000] of shard shardId-0000000000002
Mar 05, 2017 5:32:47 PM com.amazonaws.services.kinesis.clientlibrary.lib.worker.KinesisDataFetcher initialize
```

```
library(futile.logger)
library(AWR.Kinesis)

kinesis_consumer(

  initialize      = function()
    flog.info('Hello'),

  processRecords = function(records)
    flog.info(paste('Received', nrow(records), 'records from Kinesis')),

  shutdown       = function()
    flog.info('Bye'),

  updater         = list(
    list(1, function()
      flog.info('Updating some data every minute')),
    list(1/60*10, function()
      flog.info(paste(
        'This is a high frequency updater call',
        'running every 10 seconds')))),

  checkpointing = 1,
  logfile       = '/logs/logger.log')
```

Running the MultiLangDaemon locally

```
Terminix: Default
1: ec2-user@ ~
[ec2-user@ ~]$ head -n 44 logger.log
INFO [2017-03-05 03:35:23] Starting R Kinesis Consumer application
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 Start of initialize
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 Hello
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 End of initialize
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:24 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:25 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:26 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:27 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:28 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:29 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:30 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:31 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:32 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:33 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:33 UTC] shardId-000000000000 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:35:34 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:35 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:36 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:37 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:38 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:39 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:40 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:41 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:42 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:43 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:43 UTC] shardId-000000000000 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:35:44 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:45 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:46 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:47 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:48 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:49 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:50 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:51 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:52 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:53 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:54 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:54 UTC] shardId-000000000000 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:35:55 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:56 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:57 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:58 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:59 UTC] shardId-000000000000 Received 3 records from Kinesis
[ec2-user@ip logs]$
```

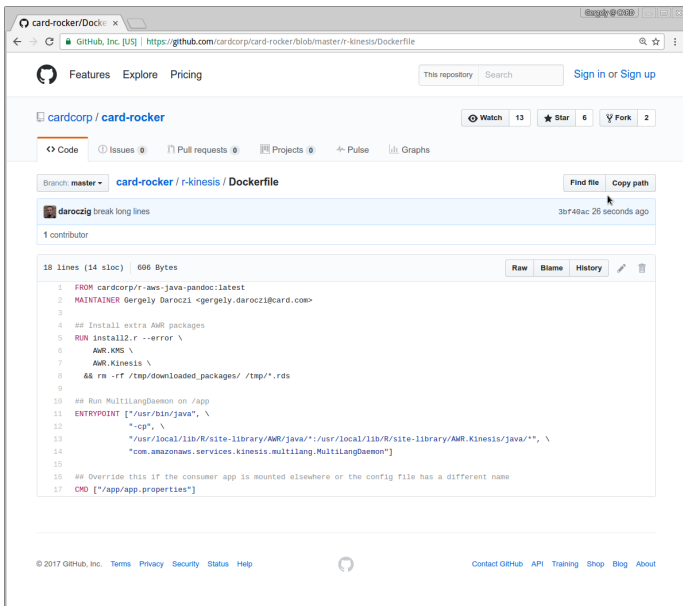
❶ Dockerize your Kinesis Consumer:

- Java
- R
- AWR, AWR.Kinesis packages
- app.R
- app.properties
- startup command

❷ Put it on Docker Hub

❸ Run as a EC2 Container Service Task:

- Create an ECS cluster
- Create ECS Task Role
- Create a Task definition
- Run it (as a service)



card-rocker/Dockerfile

cardcorp / card-rocker

Code Issues Pull requests Projects Pulse Graphs

Branch: master card-rocker / r-kinesis / Dockerfile

Find file Copy path

daroczig break long lines 3bf48ac 26 seconds ago

1 contributor

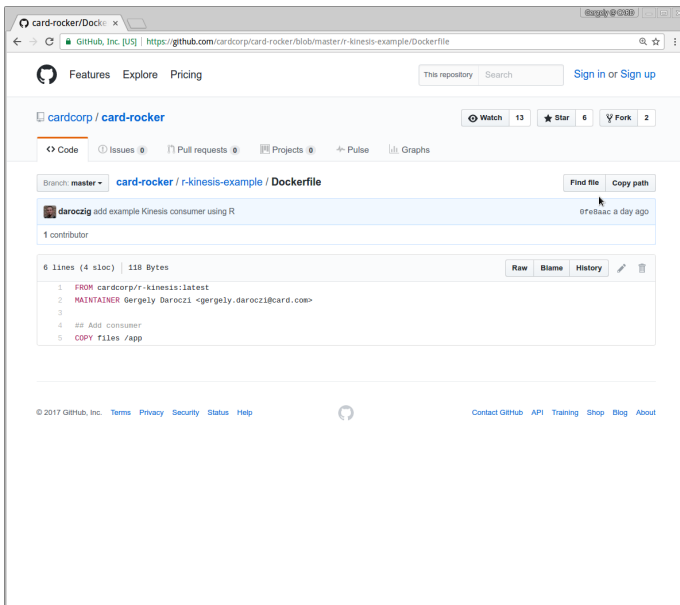
18 lines (14 sloc) | 606 Bytes

Raw Blame History

```
1 FROM cardcorp/r-aws-java-pandoc:latest
2 MAINTAINER Gergely Daroczi <gergely.daroczi@card.com>
3
4 ## Install extra AWR packages
5 RUN install2.r --error \
6     AWR_KMS \
7     AWR_Kinesis \
8     && rm -rf /tmp/downloaded_packages/ /tmp/*.rds
9
10 ## Run MultiLangDaemon on /app
11 ENTRYPOINT ["/usr/bin/java", \
12     "-cp", \
13     "/usr/local/lib/R/site-library/AMR/java/*:/usr/local/lib/R/site-library/AMR.Kinesis/java/*", \
14     "com.amazonaws.services.kinesis.multilang.MultilangDaemon"]
15
16 ## Override this if the consumer app is mounted elsewhere or the config file has a different name
17 CMD ["/app/app.properties"]
```

© 2017 GitHub, Inc. Terms Privacy Security Status Help

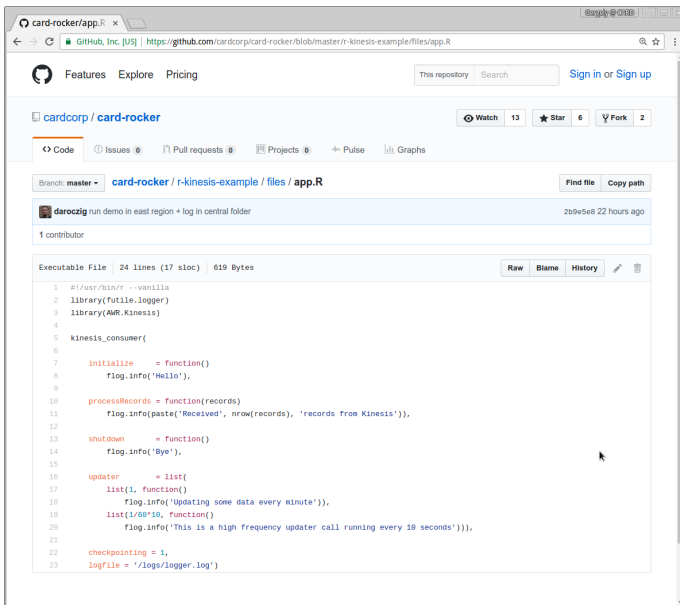
Contact GitHub API Training Shop Blog About



The screenshot shows a web browser displaying the GitHub repository page for `cardcorp / card-rocker`. The page is viewed on a desktop, with the browser address bar showing the URL `https://github.com/cardcorp/card-rocker/blob/master/r-kinesis-example/Dockerfile`. The repository page includes a header with navigation links (Features, Explore, Pricing), a search bar, and repository statistics (13 Watchers, 6 Stars, 2 Forks). The main content area shows the `Dockerfile` for the `r-kinesis-example` branch. The file was added by `daroczig` (Gergely Daroczi) a day ago. The file content is as follows:

```
1 FROM cardcorp/r-kinesis:latest
2 MAINTAINER Gergely Daroczi <gergely.daroczi@card.com>
3
4 ## Add consumer
5 COPY files /app
```

The footer of the page includes copyright information for GitHub, Inc. (© 2017), links to Terms, Privacy, Security, Status, and Help, and links to Contact GitHub, API, Training, Shop, Blog, and About.



card-rocker/app.R

GitHub, Inc. [US] | <https://github.com/cardcorp/card-rocker/blob/master/r-kinesis-example/files/app.R>

Features Explore Pricing

This repository Search Sign in or Sign up

cardcorp / card-rocker

Watch 13 Star 6 Fork 2

Code Issues 0 Pull requests 0 Projects 0 Pulse Graphs

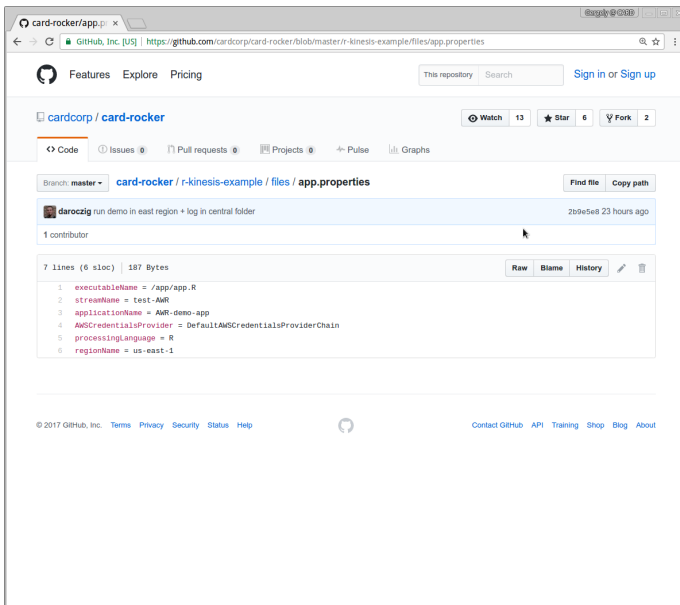
Branch: master card-rocker / r-kinesis-example / files / app.R Find file Copy path

daroczig run demo in east region + log in central folder 2b9e5e8 22 hours ago

1 contributor

Executable File 24 lines (17 sloc) 619 Bytes Raw Blame History

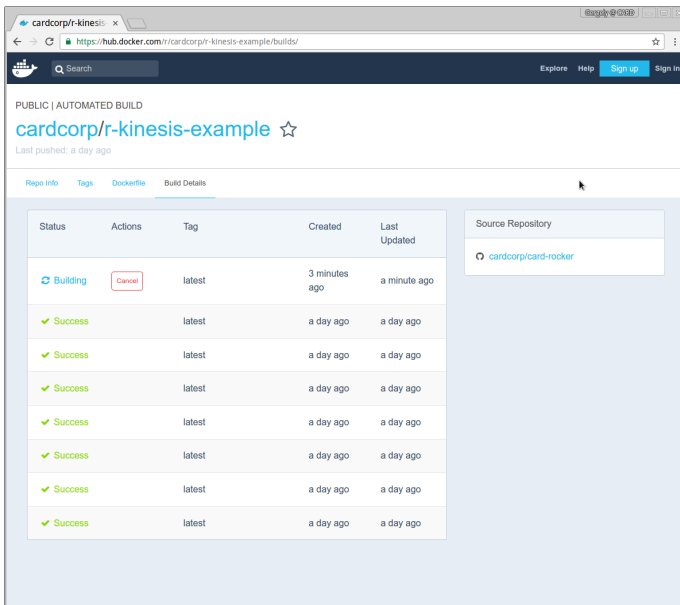
```
1 #!/usr/bin/r --vanilla
2 library(futile.logger)
3 library(AWR.Kinesis)
4
5 kinesis_consumer{
6
7   initialize = function()
8     flog.info('Hello'),
9
10  processRecords = function(records)
11    flog.info(paste('Received', nrow(records), 'records from Kinesis')),
12
13  shutdown = function()
14    flog.info('Bye'),
15
16  updater = list(
17    list(1, function()
18      flog.info('Updating some data every minute')),
19    list(1/60*10, function()
20      flog.info('This is a high frequency updater call running every 10 seconds'))),
21
22  checkpointing = 1,
23  logfile = '/logs/logger.log'}
```



The screenshot shows a web browser displaying the GitHub repository for `cardcorp / card-rocker`. The page is for the `app.properties` file in the `r-kinesis-example` directory on the `master` branch. The file content is as follows:

```
1 executableName = /app/app.R
2 streamName = test-AwR
3 applicationName = AwR-demo-app
4 AWSCredentialsProvider = DefaultAMSCredentialsProviderChain
5 processingLanguage = R
6 regionName = us-east-1
```

The page also shows a commit by `daroczig` with the message "run demo in east region + log in central folder" from 23 hours ago. The file is 187 bytes and has 7 lines of code.

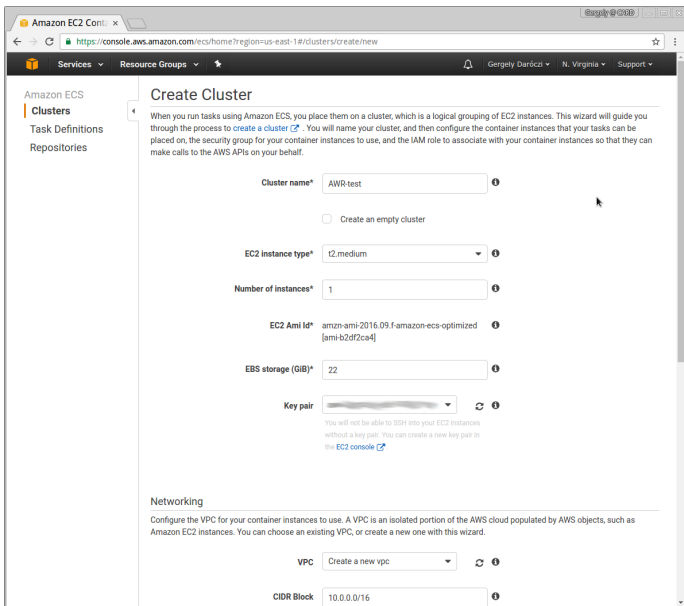


The screenshot shows the Docker Hub interface for the repository `cardcorp/r-kinesis-example`. The page is titled "PUBLIC | AUTOMATED BUILD" and shows the repository name with a star icon. Below the name, it says "Last pushed: a day ago". The "Build Details" tab is selected, showing a table of build history. The table has columns for Status, Actions, Tag, Created, and Last Updated. The first row shows a "Building" status with a "Cancel" button, while the subsequent seven rows show "Success" status. To the right of the table, the "Source Repository" is listed as `cardcorp/card-rocker`.

Status	Actions	Tag	Created	Last Updated
Building	Cancel	latest	3 minutes ago	a minute ago
Success		latest	a day ago	a day ago
Success		latest	a day ago	a day ago
Success		latest	a day ago	a day ago
Success		latest	a day ago	a day ago
Success		latest	a day ago	a day ago
Success		latest	a day ago	a day ago
Success		latest	a day ago	a day ago

Source Repository

`cardcorp/card-rocker`



The screenshot shows the 'Create Cluster' wizard in the Amazon ECS console. The left sidebar contains 'Amazon ECS', 'Clusters', 'Task Definitions', and 'Repositories'. The main content area is titled 'Create Cluster' and includes a descriptive paragraph. Below the text are several configuration fields: 'Cluster name*' (text input with 'AWR-test'), 'EC2 instance type*' (dropdown with 't2.medium'), 'Number of instances*' (text input with '1'), 'EC2 Ami Id*' (text input with 'amzn-ami-2016.09.f-amazon-ecs-optimized [ami-b2df2ca4]'), 'EBS storage (GiB)*' (text input with '22'), and 'Key pair' (dropdown menu). A note below the key pair field states: 'You will not be able to SSH into your EC2 instances without a key pair. You can create a new key pair in the [EC2 console](#)'. The 'Networking' section follows, with a descriptive paragraph and two fields: 'VPC' (dropdown with 'Create a new vpc') and 'CIDR Block' (text input with '10.0.0.0/16').

Amazon ECS

- Clusters
- Task Definitions
- Repositories

Create Cluster

When you run tasks using Amazon ECS, you place them on a cluster, which is a logical grouping of EC2 instances. This wizard will guide you through the process to [create a cluster](#). You will name your cluster, and then configure the container instances that your tasks can be placed on, the security group for your container instances to use, and the IAM role to associate with your container instances so that they can make calls to the AWS APIs on your behalf.

Cluster name* AWR-test ⓘ

☐ Create an empty cluster

EC2 instance type* t2.medium ⓘ

Number of instances* 1 ⓘ

EC2 Ami Id* amzn-ami-2016.09.f-amazon-ecs-optimized [ami-b2df2ca4] ⓘ

EBS storage (GiB)* 22 ⓘ

Key pair ⓘ

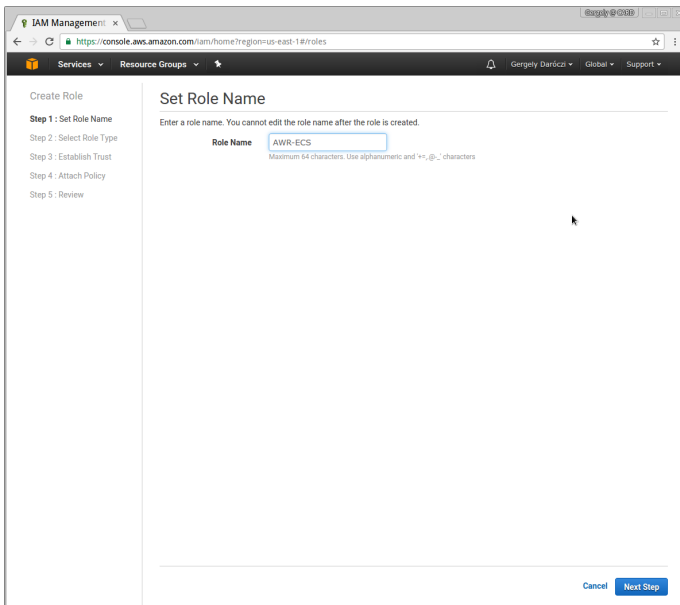
You will not be able to SSH into your EC2 instances without a key pair. You can create a new key pair in the [EC2 console](#).

Networking

Configure the VPC for your container instances to use. A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances. You can choose an existing VPC, or create a new one with this wizard.

VPC Create a new vpc ⓘ

CIDR Block 10.0.0.0/16 ⓘ



The screenshot shows the AWS IAM console interface for creating a new role. The browser address bar displays the URL `https://console.aws.amazon.com/iam/home?region=us-east-1#/roles`. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and a user profile 'Gergely Daróczy'. On the left, a 'Create Role' sidebar lists five steps: 'Step 1: Set Role Name' (selected), 'Step 2: Select Role Type', 'Step 3: Establish Trust', 'Step 4: Attach Policy', and 'Step 5: Review'. The main content area is titled 'Set Role Name' and contains the instruction 'Enter a role name. You cannot edit the role name after the role is created.' Below this, a 'Role Name' label is followed by a text input field containing 'AWR-ECS'. A small note below the input field states: 'Maximum 64 characters. Use alphanumeric and '+', '@', '_' characters'. At the bottom right of the form, there are 'Cancel' and 'Next Step' buttons.

Create Role

- Step 1: Set Role Name
- Step 2: Select Role Type
- Step 3: Establish Trust
- Step 4: Attach Policy
- Step 5: Review

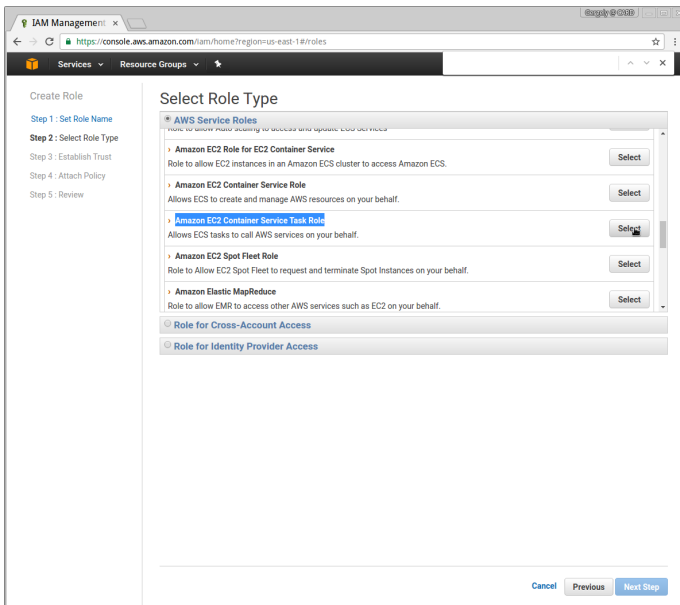
Set Role Name

Enter a role name. You cannot edit the role name after the role is created.

Role Name

Maximum 64 characters. Use alphanumeric and '+', '@', '_' characters

[Cancel](#) [Next Step](#)



The screenshot shows the AWS IAM console interface for creating a new role. The browser address bar indicates the URL is `https://console.aws.amazon.com/iam/home?region=us-east-1#/roles`. The left sidebar shows the 'Create Role' wizard with five steps: Step 1: Set Role Name, Step 2: Select Role Type (current), Step 3: Establish Trust, Step 4: Attach Policy, and Step 5: Review. The main content area is titled 'Select Role Type'. Under the 'AWS Service Roles' section, several roles are listed with 'Select' buttons. The 'Amazon EC2 Container Service Task Role' is highlighted with a blue selection bar. Below this section, there are two other role categories: 'Role for Cross-Account Access' and 'Role for Identity Provider Access'. At the bottom right of the wizard, there are three buttons: 'Cancel', 'Previous', and 'Next Step'.

Create Role

- Step 1: Set Role Name
- Step 2: Select Role Type**
- Step 3: Establish Trust
- Step 4: Attach Policy
- Step 5: Review

Select Role Type

AWS Service Roles

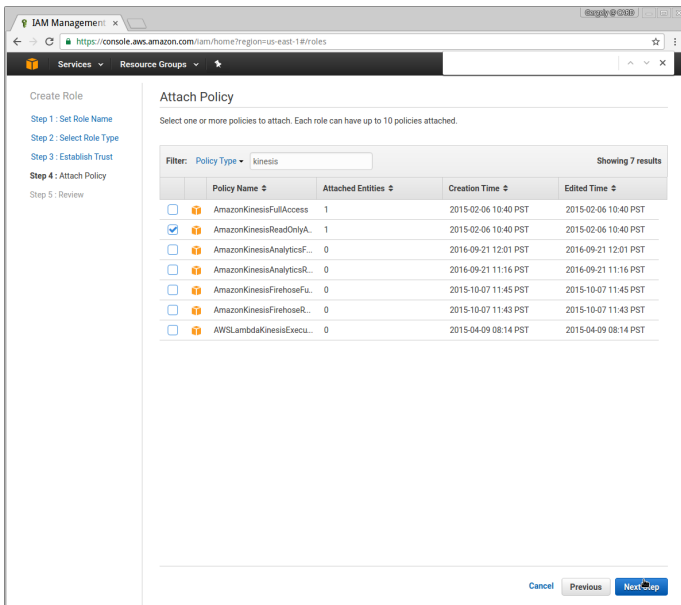
Role to allow Auto scaling to access and update EC2 services

- Amazon EC2 Role for EC2 Container Service**
Role to allow EC2 instances in an Amazon ECS cluster to access Amazon ECS. Select
- Amazon EC2 Container Service Role**
Allows ECS to create and manage AWS resources on your behalf. Select
- Amazon EC2 Container Service Task Role**
Allows ECS tasks to call AWS services on your behalf. Select
- Amazon EC2 Spot Fleet Role**
Role to Allow EC2 Spot Fleet to request and terminate Spot Instances on your behalf. Select
- Amazon Elastic MapReduce**
Role to allow EMR to access other AWS services such as EC2 on your behalf. Select

Role for Cross-Account Access

Role for Identity Provider Access

Cancel Previous Next Step



The screenshot shows the AWS IAM console interface. On the left, a sidebar contains the 'Create Role' section with five steps: 'Step 1: Set Role Name', 'Step 2: Select Role Type', 'Step 3: Establish Trust', 'Step 4: Attach Policy' (which is the active step), and 'Step 5: Review'. The main content area is titled 'Attach Policy' and includes the instruction: 'Select one or more policies to attach. Each role can have up to 10 policies attached.' Below this, there is a filter bar with 'Policy Type' set to 'kinesis' and a 'Showing 7 results' indicator. A table lists the available policies with columns for checkboxes, Policy Name, Attached Entities, Creation Time, and Edited Time. The 'AmazonKinesisReadOnlyAccess' policy is selected. At the bottom right, there are 'Cancel', 'Previous', and 'Next Step' buttons.

Create Role

- Step 1: Set Role Name
- Step 2: Select Role Type
- Step 3: Establish Trust
- Step 4: Attach Policy
- Step 5: Review

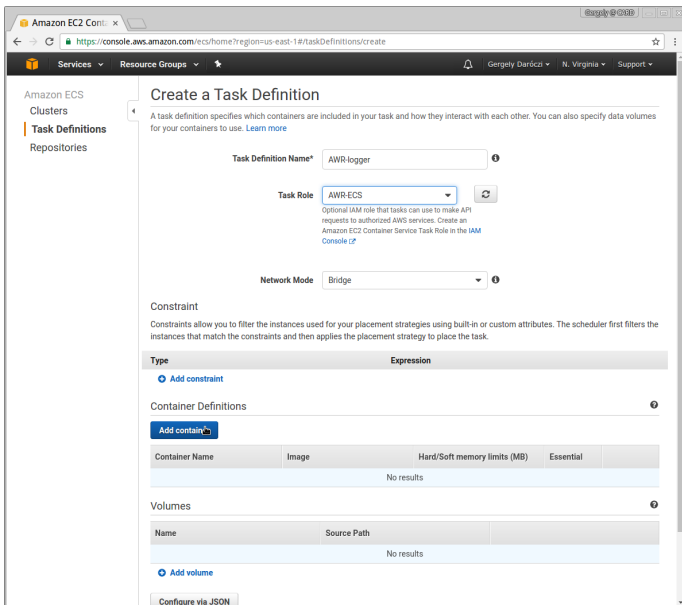
Attach Policy

Select one or more policies to attach. Each role can have up to 10 policies attached.

Filter: Policy Type Showing 7 results

	Policy Name	Attached Entities	Creation Time	Edited Time
☐	AmazonKinesisFullAccess	1	2015-02-06 10:40 PST	2015-02-06 10:40 PST
☒	AmazonKinesisReadOnlyA...	1	2015-02-06 10:40 PST	2015-02-06 10:40 PST
☐	AmazonKinesisAnalyticsF...	0	2016-09-21 12:01 PST	2016-09-21 12:01 PST
☐	AmazonKinesisAnalyticsR...	0	2016-09-21 11:16 PST	2016-09-21 11:16 PST
☐	AmazonKinesisFirehoseFu...	0	2015-10-07 11:45 PST	2015-10-07 11:45 PST
☐	AmazonKinesisFirehoseR...	0	2015-10-07 11:43 PST	2015-10-07 11:43 PST
☐	AWSLambdaKinesisExecu...	0	2015-04-09 08:14 PST	2015-04-09 08:14 PST

Cancel Previous **Next Step**



Amazon EC2 Cont... x

← → ↻ <https://console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/create> ☆

Services ▾ Resource Groups ▾

Alerts Gergely Daróczi ▾ N. Virginia ▾ Support ▾

Amazon ECS
Clusters
Task Definitions
Repositories

Create a Task Definition

A task definition specifies which containers are included in your task and how they interact with each other. You can also specify data volumes for your containers to use. [Learn more](#)

Task Definition Name* ⓘ

Task Role ⓘ

Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon EC2 Container Service Task Role in the [IAM Console](#) ⓘ

Network Mode ⓘ

Constraint

Constraints allow you to filter the instances used for your placement strategies using built-in or custom attributes. The scheduler first filters the instances that match the constraints and then applies the placement strategy to place the task.

Type	Expression
Add constraint	

Container Definitions ⓘ

[Add container](#)

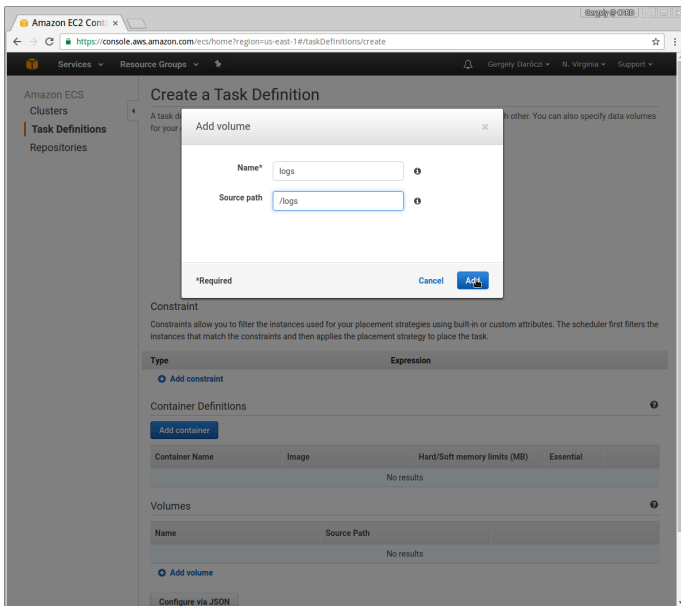
Container Name	Image	Hard/Soft memory limits (MB)	Essential
No results			

Volumes ⓘ

Name	Source Path
No results	

[Add volume](#)

[Configure via JSON](#)



The screenshot shows the Amazon ECS console interface for creating a task definition. A modal window titled "Add volume" is open, showing a form with the following fields:

- Name***: logs
- Source path**: /logs

Below the modal, the "Constraint" section is visible, explaining that constraints filter instances based on attributes. Below that, the "Container Definitions" section has an "Add container" button. At the bottom, the "Volumes" section has an "Add volume" button. The "Add volume" button is highlighted with a blue border.

Constraint

Constraints allow you to filter the instances used for your placement strategies using built-in or custom attributes. The scheduler first filters the instances that match the constraints and then applies the placement strategy to place the task.

Type **Expression**

[Add constraint](#)

Container Definitions

[Add container](#)

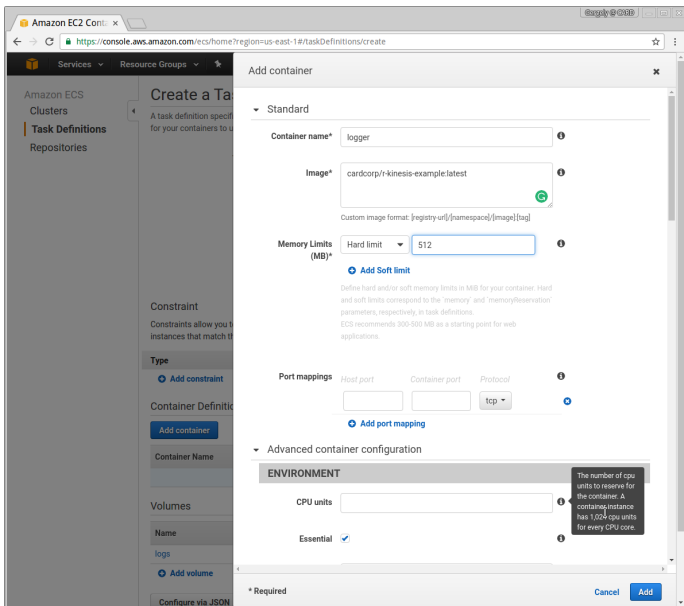
Container Name	Image	Hard/Soft memory limits (MB)	Essential
No results			

Volumes

Name	Source Path
No results	

[Add volume](#)

[Configure via JSON](#)



Amazon EC2 Cont... x

Services Resource Groups

Amazon ECS
Clusters
Task Definitions
Repositories

Create a Task Definition

A task definition specifies the metadata for containers that you want to run on Amazon ECS.

Constraint

Constraints allow you to specify the resources that match the task definition.

Type

Add constraint

Container Definition

Add container

Container Name

Volumes

Name
logs

Add volume

Configure via JSON

Add container

Standard

Container name* logger

Image* cardcorp/r-kinesis-example:latest

Custom image format: {registry-url}/{namespace}/{image}:{tag}

Memory Limits (MB)*

Hard limit 512

Add Soft limit

Define hard and/or soft memory limits in MiB for your container. Hard and soft limits correspond to the 'memory' and 'memoryReservation' parameters, respectively, in task definitions. ECS recommends 300-500 MB as a starting point for web applications.

Port mappings

Host port	Container port	Protocol
		tcp

Add port mapping

Advanced container configuration

ENVIRONMENT

CPU units

Essential ☒

The number of cpu units to reserve for the container. A container instance has 1,024 cpu units for every CPU core.

* Required

Cancel Add

Create a Task definition

Amazon EC2 Cont... x

← → ↻ <https://console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/create> ☆ ⋮

Services ▾ Resource Groups ▾

Amazon ECS
Clusters
Task Definitions
Repositories

Create a Task Definition

A task definition specifies the container images and other configuration for your containers to use.

Constraint

Constraints allow you to specify the instances that match the task definition.

Type

➤ Add constraint

Container Definition

➤ Add container

Container Name

Volumes

Name

logs

➤ Add volume

Configure via JSON

Add container

IP address

➤ Add extra host

STORAGE AND LOGGING

Read only root file system ☐

Mount points

Source volume logs

Container path /logs

Read only ☐

➤ Add mount point

Volumes from

Source container

Read only ☐

➤ Add volumes

Log configuration

Log driver <none>

Log options

Key Add key

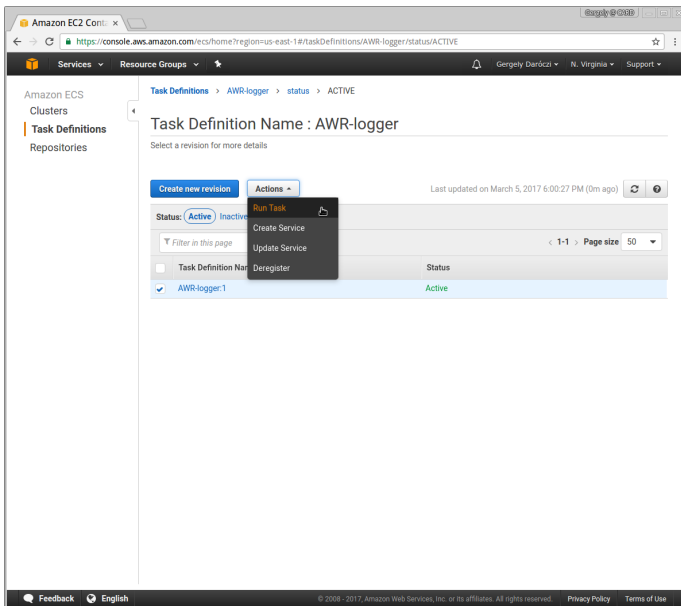
Value Add value

SECURITY

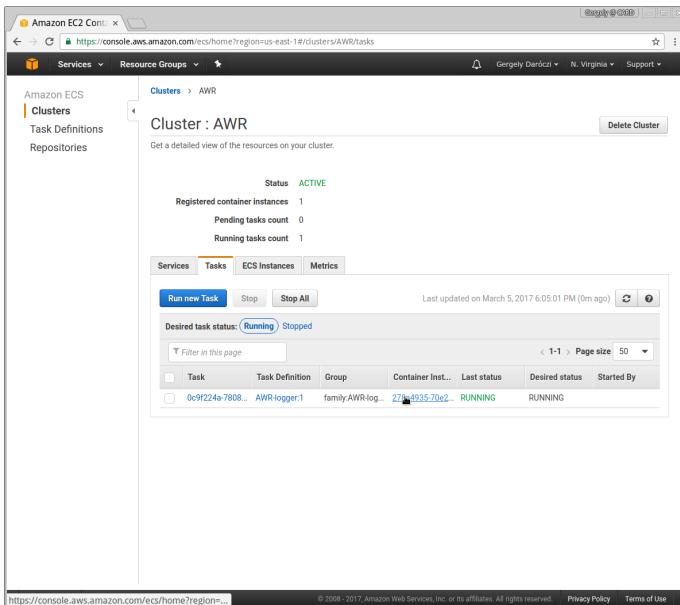
* Required

Cancel Add

When this parameter is true, the container is given read-only access to its root file system.



The screenshot shows the Amazon ECS console interface. The breadcrumb navigation is **Task Definitions > AWR-logger > status > ACTIVE**. The main heading is **Task Definition Name : AWR-logger**, with a subtext **Select a revision for more details**. On the left sidebar, **Task Definitions** is selected. The main content area shows a table of task definitions. The first row, **AWR-logger:1**, is selected and highlighted in blue, with a status of **Active**. An **Actions** dropdown menu is open over the selected row, showing options: **Run Task** (highlighted in orange), **Create Service**, **Update Service**, and **Deregister**. Above the table, there is a **Create new revision** button and a **Last updated on March 5, 2017 6:00:27 PM (0m ago)** timestamp. The footer of the console shows **Feedback**, **English**, and copyright information for Amazon Web Services, Inc. (© 2008 - 2017).



Amazon EC2 Console

Services Resource Groups

Amazon ECS

- Clusters
- Task Definitions
- Repositories

Clusters > AWR

Cluster : AWR

Get a detailed view of the resources on your cluster.

Status **ACTIVE**

Registered container instances 1

Pending tasks count 0

Running tasks count 1

Services Tasks ECS Instances Metrics

Run new Task Stop Stop All

Last updated on March 5, 2017 6:05:01 PM (0m ago)

Desired task status: **Running** Stopped

Filter in this page

< 1-1 > Page size 50

	Task	Task Definition	Group	Container Inst...	Last status	Desired status	Started By
☐	0c9f224a-7808...	AWR-logger:1	family:AWR-log...	27b-4935-70e2...	RUNNING	RUNNING	

https://console.aws.amazon.com/ecs/home?region=... © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

```
Terminix: Default

1: ec2-user@ ~: ~
/ ssh -i ~/ssh/ ec2-user@
Last login: Mon Mar 6 02:05:29 2017 from

_ _ _ _ _
| | | | |
|_|(|_|_|_|_|

Amazon ECS-Optimized Amazon Linux AMI 2016.09.f

For documentation visit, http://aws.amazon.com/documentation/ecs
4 package(s) needed for security, out of 6 available
Run "sudo yum update" to apply all updates.
[ec2-user@ ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED            STATUS             PORTS
b59981414dda        cardcorp/r-kinesis-example:latest  "/usr/bin/java -cp /u"  22 hours ago       Up 22 hours
cs-AWR-logger-1-logger-92ca888bd5d681e59d01  amazon/amazon-ecs-agent:latest     "/agent"              23 hours ago       Up 23 hours
[ec2-user@ ~]$ pgrep app.R
29435
[ec2-user@ ~]$ head -n 10 /logs/logger.log
INFO [2017-03-05 03:35:23] Starting R Kinesis Consumer application
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 Start of initialize
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 Hello
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 End of initialize
INFO [2017-03-05 03:35:23 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:24 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:25 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:26 UTC] shardId-000000000000 Received 2 records from Kinesis
INFO [2017-03-05 03:35:27 UTC] shardId-000000000000 Received 3 records from Kinesis
INFO [2017-03-05 03:35:28 UTC] shardId-000000000000 Received 2 records from Kinesis
[ec2-user@ ~]$
```

Scaling the Kinesis Consumer up

```
Terminix: Default
1: ec2-user@ ~ -
INFO [2017-03-05 03:43:01 UTC] shardId-000000000000 Received 1 records from Kinesis
INFO [2017-03-05 03:43:01 UTC] shardId-000000000000 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:43:09 UTC] shardId-000000000000 Shutting down
INFO [2017-03-05 03:43:09 UTC] shardId-000000000000 Bye
INFO [2017-03-05 03:43:15] Starting R Kinesis Consumer application
INFO [2017-03-05 03:43:15 UTC] shardId-000000000002 Start of initialize
INFO [2017-03-05 03:43:15 UTC] shardId-000000000002 Hello
INFO [2017-03-05 03:43:15 UTC] shardId-000000000002 End of initialize
INFO [2017-03-05 03:43:15] Starting R Kinesis Consumer application
INFO [2017-03-05 03:43:16 UTC] shardId-000000000001 Start of initialize
INFO [2017-03-05 03:43:16 UTC] shardId-000000000001 Hello
INFO [2017-03-05 03:43:16 UTC] shardId-000000000001 End of initialize
INFO [2017-03-05 03:44:32 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:32 UTC] shardId-000000000002 Updating some data every minute
INFO [2017-03-05 03:44:32 UTC] shardId-000000000002 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:44:33 UTC] shardId-000000000002 Received 3 records from Kinesis
INFO [2017-03-05 03:44:34 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:35 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:36 UTC] shardId-000000000001 Received 2 records from Kinesis
INFO [2017-03-05 03:44:36 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:36 UTC] shardId-000000000001 Updating some data every minute
INFO [2017-03-05 03:44:36 UTC] shardId-000000000001 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:44:37 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:38 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:39 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:39 UTC] shardId-000000000001 Received 2 records from Kinesis
INFO [2017-03-05 03:44:40 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:40 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:41 UTC] shardId-000000000001 Received 2 records from Kinesis
INFO [2017-03-05 03:44:42 UTC] shardId-000000000001 Received 2 records from Kinesis
INFO [2017-03-05 03:44:43 UTC] shardId-000000000002 Received 2 records from Kinesis
INFO [2017-03-05 03:44:43 UTC] shardId-000000000002 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:44:44 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:45 UTC] shardId-000000000002 Received 2 records from Kinesis
INFO [2017-03-05 03:44:45 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:46 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:46 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:47 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:47 UTC] shardId-000000000001 This is a high frequency updater call running every 10 seconds
INFO [2017-03-05 03:44:47 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:48 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:48 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:49 UTC] shardId-000000000001 Received 1 records from Kinesis
INFO [2017-03-05 03:44:49 UTC] shardId-000000000002 Received 1 records from Kinesis
INFO [2017-03-05 03:44:50 UTC] shardId-000000000001 Received 1 records from Kinesis
```

Nice example project, but ...

- I might want to avoid publishing my Consumer on Docker Hub
- I might want to avoid publishing my code on GitHub
- I might want to avoid committing credentials etc to the repo

Problems:

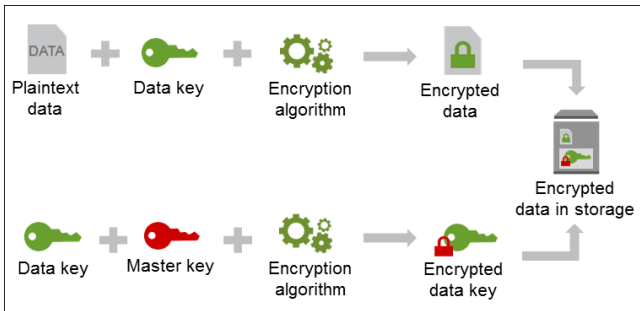
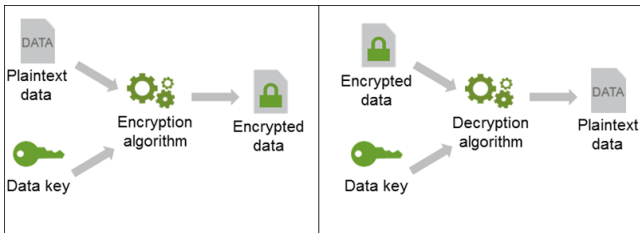
- How to store credentials in the Docker images?
- Where to store the Docker images?

Nice example project, but ...

- I might want to avoid publishing my Consumer on Docker Hub
- I might want to avoid publishing my code on GitHub
- I might want to avoid committing credentials etc to the repo

Problems:

- How to store credentials in the Docker images? **KMS**
- Where to store the Docker images? **ECR**



Source: [AWS Encryption SDK](#)

- encrypt up to 4 KB of arbitrary data:

```
> library(AWR.KMS)
> kms_encrypt('alias/mykey', 'foobar')
[1] "Base-64 encoded ciphertext"
```

- decrypt such Base-64 encoded ciphertext back to plaintext:

```
> kms_decrypt('Base-64 encoded ciphertext')
[1] "foobar"
```

- generate a data encryption key:

```
> kms_generate_data_key('alias/mykey')
$cipher
[1] "Base-64 encoded, encrypted data encryption key"
$key
[1] "alias/mykey"
$text
[1] 00 01 10 11 00 01 10 11 ...
```

```
## let's say we want to encrypt the mtcars dataset stored in JSON
library(jsonlite)
data <- toJSON(mtcars)

## generate a 256-bit data encryption key (that's supported by digest::AES)
library(AWR.KMS)
key <- kms_generate_data_key('alias/mykey', byte = 32L)

## convert the JSON to raw so that we can use that with digest::AES
raw <- charToRaw(data)
## the text length must be a multiple of 16 bytes
## https://github.com/sdoyen/r_password_crypt/blob/master/crypt.R
raw <- c(raw, as.raw(rep(0, 16 - length(raw) %% 16)))

## encrypt the raw object with the new key + digest::AES
## the resulting text and the encrypted key can be stored on disk
library(digest)
aes <- AES(key$text)
base64_enc(aes$encrypt(raw))

## decrypt the above returned ciphertext using the decrypted key
rawToChar(aes$decrypt(base64_dec(...), raw = TRUE))
```

```
library(AWR.Kinesis); library(jsonlite); library(AWR.KMS); library(futile.logger); flog.threshold(DEBUG)

kinesis_consumer(
  initialize      = function() {
    flog.info('Decrypting Redis hostname via KMS')
    host <- kms_decrypt('AQECAHiiz4GEPFQLL9AAON5TY/1DR5euQQScpXQU9iYTn+u...')
    flog.info('Connecting to Redis')
    library(rredis); redisConnect(host = host)
    flog.info('Connected to Redis')
  },
  processRecords = function(records) {
    flog.info(paste('Received', nrow(records), 'records from Kinesis'))
    for (record in records$data) {
      flight <- fromJSON(record)$dest
      if (!is.null(flight)) {
        flog.debug(paste('Adding +1 to', flight))
        redisIncr(sprintf('flight:%s', flight))
      } else {
        flog.error('Flight destination not found')
      }
    }
  },
  updater = list(
    list(1/6, function() {
      flog.info('Checking overall counters')
      flights <- redisKeys('flight:*')
      for (flight in flights) {
        flog.debug(paste('Found', redisGet(flight), sub('^flight:', '', flight)))
      }
    })),
  logfile = '/logs/redis.log')
```

Dockerfile:

```
FROM cardcorp/r-kinesis:latest
MAINTAINER Gergely Daroczi <gergely.daroczi@card.com>

## Install R package to interact with Redis
RUN install2.r --error rredis && rm -rf /tmp/downloaded_packages/ /tmp/*.rds

## Add consumer
COPY files /app
```

Build and push to ECR:

```
docker build -t cardcorp/r-kinesis-secret .
`aws ecr get-login --region us-east-1`
docker tag -f cardcorp/r-kinesis-secret:latest \
    ***.dkr.ecr.us-east-1.amazonaws.com/cardcorp/r-kinesis-secret:latest
docker push ***.dkr.ecr.us-east-1.amazonaws.com/cardcorp/r-kinesis-secret:latest
```

```
library(treemap);library(highcharter);library(nycflights13)
library(rredis);redisConnect(host = '***', port = '***')

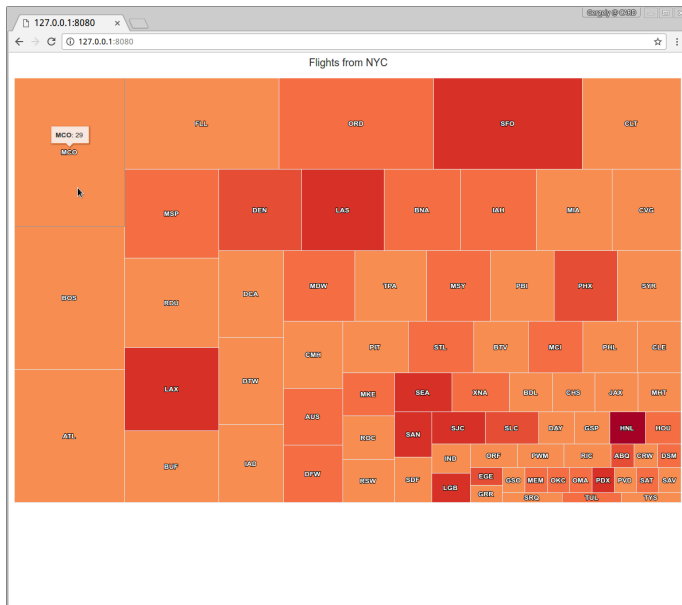
ui      <- shinyUI(highchartOutput('treemap', height = '800px'))
server  <- shinyServer(function(input, output, session) {

  destinations <- reactive({
    reactiveTimer(2000)()
    flights <- redisMGet(redisKeys('flight:*'))
    flights <- data.frame(faa = sub('^flight:', '', names(flights)),
                          N   = as.numeric(flights))
    merge(flights, airports, by = 'faa')
  })

  output$treemap <- renderHighchart({
    tm <- treemap(destinations(), index = c('faa'),
                  vSize = 'N', vColor = 'tz',
                  type = 'value', draw = FALSE)
    hc_title(hctreemap(tm, animation = FALSE), text = 'Flights from NYC')
  })

})

shinyApp(ui = ui, server = server)
```



- AWR repo:
 - 6.3 GB
 - 273 tags/versions
 - GitLab + CI + drat

```
install.packages('AWR', repos = 'https://cardcorp.gitlab.io/AWR')
```

- AWR repo:
 - 6.3 GB
 - 273 tags/versions
 - GitLab + CI + drat

```
install.packages('AWR', repos = 'https://cardcorp.gitlab.io/AWR')
```

- Submitted to CRAN on
 - 2016-12-05

- AWR repo:
 - 6.3 GB
 - 273 tags/versions
 - GitLab + CI + drat

```
install.packages('AWR', repos = 'https://cardcorp.gitlab.io/AWR')
```

- Submitted to CRAN on
 - 2016-12-05
 - 2017-01-09
 - 2017-01-10
 - 2017-01-11
 - 2017-01-11
 - 2017-01-13

- AWR repo:
 - 6.3 GB
 - 273 tags/versions
 - GitLab + CI + drat

```
install.packages('AWR', repos = 'https://cardcorp.gitlab.io/AWR')
```

- Submitted to CRAN on
 - 2016-12-05
 - 2017-01-09
 - 2017-01-10
 - 2017-01-11
 - 2017-01-11
 - 2017-01-13
- Release cycle: 2 minor, ~125 patch versions in the past 12 months
- CI



```
> library(rJava)
> kc <- .jnew('com.amazonaws.services.s3.AmazonS3Client')
> kc$getS3AccountOwner()$getDisplayName()
[1] "foobar"
```

